# Evolving GP classifiers for streaming data tasks with concept change and label budgets: A benchmarking study*

Ali Vahdat, Jillian Morgan, Andrew R. McIntyre,
Malcolm I. Heywood and Nur Zincir-Heywood†

April 21, 2016

**Abstract**

Streaming data classification requires that several additional challenges are addressed that are not typically encountered in offline supervised learning formulations. Specifically, access to data at any training generation is limited to a small subset of the data, and the data itself is potentially generated by a non-stationary process. Moreover, there is a cost to requesting labels, thus a label budget is enforced. Finally, an anytime classification requirement implies that it must be possible to identify a 'champion' classifier for predicting labels as the stream progresses. In this work, we propose a general framework for deploying genetic programming (GP) to streaming data classification under these constraints. The framework consists of a sampling policy and an archiving policy that enforce criteria for selecting data to appear in a data subset. Only the exemplars of the data subset are labeled, and it is the content of the data subset that training epochs are performed against. Specific recommendations include support for GP task decomposition / modularity and making additional training epochs per data subset. Both recommendations make significant improvements to the baseline performance of GP under streaming data with label budgets. Benchmarking issues addressed include the identification of datasets and performance measures.

## 1 Introduction

A traditional view of learning from data is most often characterized by the supervised learning 'classification' task. However, as we are increasingly encountering data rich environments, the basis for such a characterization are becoming less relevant. Decision making from streaming data is one such application area (e.g., stock market data, utility utilization, behavioural modelling, sentiment analysis, process monitoring etc.). Under a 'streaming' scenario for constructing models of classification, data arrives on a continuous basis, thus there is no concept of a 'beginning' or an 'end'. It is not possible to see all the data at once and it therefore becomes impossible to guarantee that the data 'seen' at any point in time

*This technical report is prepared for compliance to NSERC open source publication requirements. The original article should be referenced in all cases: A. Vahdat et al. (2015) Chapter 18 in ***Handbook of Genetic Programming Applications***. A. H. Gandomi et al. (eds) Springer. pp 451–480

†Faculty of Computer Science, Dalhousie University

are representative of the 'whole' task. Indeed, the process generating the data are frequently non-stationary. Applications display properties such as concept drift (a gradual change in the process creating the data) or concept shift (sudden changes to the process creating the data). Concept change in general implies that a model that functions effectively at one point in the stream will not necessarily function effectively later on. Moreover, in the general case it is not possible to provide labels for all the stream. Instead decisions need to be made regarding what to label without calling upon an oracle. Indeed, the real-time nature of many streaming classification tasks implies that the number of label requests needs to be very much lower than the throughput of the stream itself. When combined with the issue of non-stationarity, this makes it much more difficult to recognize when models need to be rebuilt. Finally, we note that an 'anytime' nature to the task exists. Irrespective of the state of the model building process itself, a champion individual (classifier) must be available to suggest labels for the current content of the data stream at any given time.

A distinction is drawn between regression (function approximation) and classification under streaming data. Regression under a streaming data context is most synonymous with the task of forecasting. As such the true value for the dependent variable is known a short time after a prediction is made by the model. This means that the issue of label budgets is not as prevalent, and models can therefore be much more reactive. Conversely, having to explicitly address the issue of label budgets implies that independent processes need to be introduced to prompt for label information (e.g. change detection).

Various proposals have been made for what properties GP might need to assume under environments that are in some way 'dynamic'. Several researchers have made a case for adopting modular frameworks for model building under dynamic scenarios. For example, environments that change their objective dynamically over the course of evolution (e.g., [26]). Likewise, modularity might be deemed useful from the perspective of delimiting the scope of variation operators, thus making credit assignment more transparent and facilitating incremental modification (e.g., [43]). Diversity maintenance represents a reoccurring theme, and is frequently emphasized by research in (non-evolutionary) ensemble learning frameworks applied to streaming tasks [9, 33, 37]. Finally, we note that 'evolvability' is typically defined in terms of a combination of the ability to support (phenotypic) variability and the fitness of the resulting offspring (in a future environment) (e.g., [34]). This can be viewed through the perspective of Baldwin mechanisms for evolution. Thus, it is desirable to retain parents that are most likely to lead to fit offspring on a regular basis.

This work undertakes a benchmarking study of a framework previously proposed for evolving modular GP individuals under streaming data contexts with label budgets [40]. The specific form of GP assumed takes the form of Symbiotic Bid-Based GP (SBB) and is hereafter denoted **StreamSBB**. The framework consists of three elements: a sampling policy, a data subset, and a data archiving policy. The combination of sampling policy and the data subset achieve a decoupling between the rate at which the stream passes and the rate at which evolution commences. This also provides the basis for changing the distribution of data from that present in the stream at any point in time. In changing the distribution of data, we are in a position to, for example, resist the impact of class imbalance. Finally, in order to address the issue of model building under a limited **label budget**, a stochastic querying scheme is assumed. Thus, for any given window location, a fixed number of label requests are permitted. The selection of an appropriate label budget being a reflection of the cost of making label requests.

Benchmarking will be performed with both artificially created datasets (which provide the ability to embed known forms of concept drift and shift) as well as real-world datasets (electricity utilization / demand and forest cover types). Such datasets display a wide range of real world properties, with cardinality measured close to the millions, dummy attributes, class imbalance, and changing relationships between attribute and label. Moreover, benchmarking practices for streaming data are explicitly identified. In particular rather than assuming a 'prequential' accuracy metric, a formulation of (average) multi-class detection rate is assumed and estimated incrementally. This enables us to avoid the caveats that appear with accuracy style metrics under class imbalance. Comparator algorithms are included from the MOA toolbox, representing current state of the art in non-evolutionary approaches to streaming data classification. The benchmarking study demonstrates the appropriateness of assuming the StreamSBB framework, with specific recommendations made regarding the utility of: modularity, pre-training, and generations per sample of labelled data.

Section 2 provides a summary of related streaming data research. The StreamSBB framework is detailed in Section 3 with the experimental methodology discussed in Section 4. Section 5 presents results of the benchmarking study where this is designed to illustrate the contribution from various components of StreamSBB. Section 6 discusses the resulting findings and concludes the work.

## 2 Related Work

A significant body of work has developed regarding the application of machine learning (ML) to various streaming classification tasks [35, 19, 5, 20, 24]. For brevity we concentrate on the issue of change detection which lies at the centre of building ML frameworks capable of operating under label budgets. Indeed, classification under label budgets represents the most recent trend in streaming data classification. We identify three broad categories of interest, outlined as follows:

**Properties specific to the model of classification** imply that measurements specific to an ML framework are made and compared to a prior characterization. For example, changes to the frequency of leaf node utility in decision trees might signify change, thus trigger label requests [17, 25].

**Properties of the input data** imply that change detection now focuses on characterizing behaviour relative to sliding window content. The principle design decision is with regards to what statistic to adopt. For example, Chernoff bounds [27], entropy [10, 42], Kullback–Leibler divergence [36], Hoeffding bounds [6], Fractal correlation dimension [18] or Hellinger divergence metric [14]. Potential drawbacks of pursuing such an approach include: 1) it is often necessary to label the data (i.e., metrics are estimated class-wise); and 2) changes to the association between label and input are not detected [39].

**Properties of the label space** imply that the classifier output 'behaviour' is quantified. For example, statistical characterizations of class boundary information (cf. classifier confidence) have been proposed [31, 38]. Thus, thresholds might be used to detect changes in classifier certainly [32], or changes to the number of confident predictions [28].

However, none of the above approaches are able to detect when a previously encountered input, $P(x)$, is associated with a new or different class label. Thus, under this scenario a label space characterization would still associate $P(x)$ with the previous label. Likewise change detection based on an input data formulation would not register any change either, i.e. $P(x)$ has not changed. Under these scenarios generating label requests uniformly (i.e., independently of the measurable properties) has been shown to be surprisingly effective [44]; as have hybrid approaches combining label space and uniform sampling [39].

Under the guise of evolutionary computation (EC) in general, a body of research has been developed regarding dynamic optimization tasks (e.g., [8]). However, such tasks are distinct from streaming data classification in that the emphasis is with regards to tracking and identifying multiple optima; thus, there is no concept of operating under label budgets.

From a genetic programming (GP) perspective, most developments come with respect to the specifics of evolving trading agents for financial applications (e.g., [13]). Although change detection is certainly important to building effective trading agents, the rebuilding of models is either performed on a continuous basis (as in function approximation) [13] or incrementally based on task specific properties such as an unacceptable loss [24]. Thus, the issue of label budgets does not appear in frameworks for evolving trading agents. Finally, we note that in the special case of learning classifier systems (LCS), an explicitly online variant has been proposed in which probabilistic heuristics are used to fill 'gaps' in the provision of label information [4].

## 3 Methodology

Figure 1 provides a summary of the general architecture assumed for applying GP to streaming data under finite labelling budgets [40, 24]. We assume a non-overlapping 'sliding window' as the generic interface to the stream. For a given window location a fixed number of samples are taken. Let $SW(i)$ denote the location of the window and 'gap' denote the data sampled from this location, or $Gap(i) \in SW(i)$; where $|Gap(i)| \leq |SW(i)|$. The sampling policy determines which exemplars are selected to appear in $Gap(i)$ for a given window location $SW(i)$. Note that it is only the $|Gap(i)|$ exemplars chosen that have their corresponding labels requested and are added to the Data Subset ($DS(i)$). An archiving policy determines which exemplars are replaced from DS at each update. Note also that the rate at which GP training epochs are performed, $gen(k)$, is a function of the rate at which DS is updated or $j = i \times k; k \in \{1, 2, ...\}$. This means that for each update in DS content (index $i$), at least a single GP training epoch is performed ($k = 1$). Naturally, increasing the number of training epochs potentially increases the capacity to react to changes to stream content, but may potentially result in over learning (w.r.t. current DS content). Hereafter we will refer to this as **DS oversampling**. Section 5 will explicitly investigate this property in more detail.

The StreamSBB framework adopts symbolic bid-based GP (SBB) as the GP architecture [16]. Specifically, SBB evolves teams of bid-based GP individuals to cooperatively decompose the classification task without having to specify team size. Supporting modularity in general has been deemed to be useful for dynamic task environments (Section 2), a property we explicitly verify in Section 5. Secondly, SBB assumes a Pareto archiving policy with diversity maintenance heuristics for enforcing a finite archive size. In effect, the concept of Pareto dominance is used to identify exemplars for retaining within DS. As such this gives them a 'lifetime' beyond the current location of the sliding window.

### 3.1 Sampling Policy

Rather than evaluating a GP classifier with respect to all data within $SW(i)$, a sampling policy is assumed to control entry into the Data Subset $(DS(i))$. This decouples the cost of any single training epoch and enforces the labelling budget, i.e. we control the cardinality of the data subset, but cannot control the throughput of the stream. Note, however, that the decision regarding the sampling of 'gap' exemplars from sliding window location $SW(i)$ to a data subset can only be performed *without* label information. It is only after identifying the exemplars included in $Gap(t)$ that labels are requested.

Two basic approaches for defining sampling policies have been identified in the wider literature (Section 2): stochastic sampling or classification confidence information. Classifier confidence information implies that as the certainty of the class label suggested by a classifier decreases (i.e. approaches ambiguity), then a label request is made [32, 39]. In the case of

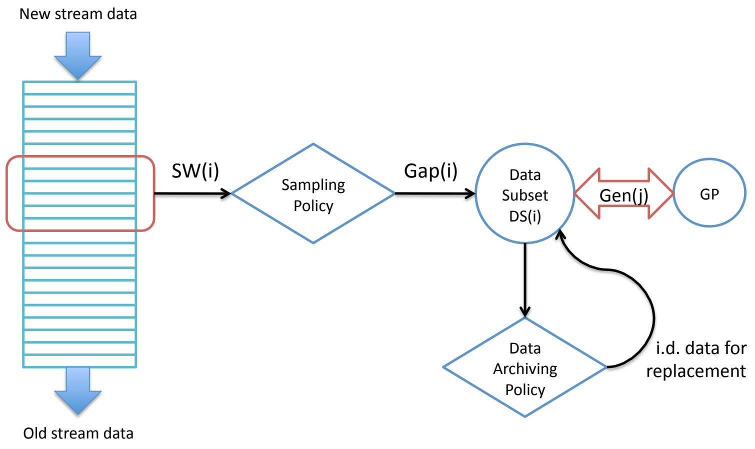

Figure 1: Components of generic architecture for applying GP to streaming data under a label budget

stochastic label requests, this is performed uniformly relative to exemplars that are classified with certainty. The objective being to confirm that cases which are classified with certainty have not undergone some form of shift into a different class. Moreover, we also note that even requesting labels with a uniform probability (under a label budget) is often better than more sophisticated heuristics [44]. In this work we will assume the uniform sampling heuristic under a label budget.

The specific form of GP assumed takes the form of Symbiotic Bid-Based GP (SBB) and therefore benefits from the ability to perform task decomposition (construct a classifier as a team of programs). Aside from the additional transparency of the resulting solutions, pursuing a GP teaming approach also provides an elegant solution to multi-class classification. A short description of SBB is provided in Section 3.4, whereas readers are referred to the earlier papers for further details [29, 16].

## 3.2 Data Archiving Policy

The scheme assumed for prioritizing DS content for replacement is defined by a **data archiving policy**. Specifically, Pareto archiving is used to identify exemplars that 'distinguish' between the performance of GP classifiers. Such a set of exemplars are said to be non-dominated [12]. One of the drawbacks of assuming a Pareto archiving policy, however, is that the archive of exemplars distinguishing between different GP classifiers increases to $P^2 - P$; where $P$ is the size of the GP population. This would have implications for the overall efficiency of the algorithm. Hence, we limit the size of DS to a suitable finite value and employ a DS exemplar diversity / aging heuristic [1]. The process for choosing exemplars from DS for replacement switches between the following cases, depending on which condition is satisfied. Let the exemplars from DS forming 'distinctions' be $d$ and those not supporting a distinction be $\bar{d}$:

**Case 1** *The number of exemplars forming a distinction is less than or equal to* $|DS| - |Gap|$ (i.e. $|d| \leq |DS| - |Gap|$). This implies that the number of exemplars that *do not* support distinctions is greater than or equal to $|Gap|$ (i.e. $|\bar{d}| \geq |Gap|$). Hence, the DS exemplars replaced by $Gap(i)$ are selected from $\bar{d}$ alone.

**Case 2** *The number of exemplars forming distinctions is more than* $|DS| - |Gap|$. Any exemplars not forming distinctions ($\bar{d}$) will be replaced. In addition $|Gap| - |\bar{d}|$ exemplars forming distinctions will also be replaced, potentially resulting in the loss of GP classifiers (i.e., no longer identified as being non-dominated). The exemplars forming distinctions are now ranked in accordance with how many other points form the same distinction and how long an exemplar has been in the archive [1]. In effect exemplars supporting: 1) unique distinctions see more priority than those forming more common distinctions i.e., a form of fitness sharing or diversity maintenance, and 2) older exemplars are more likely to removed in favour of those forming more recent distinctions.

Such preference schemes were previously shown to be useful under GP streaming classification, albeit without label budgeting [2, 1]. Further details of Pareto archiving as applied to GP are available in [16].

## 3.3 Anytime Classifier Operation

In order to predict class labels for exemplars of the non-stationary stream a single GP individual must be present at any point in time to perform this task, or **anytime classifier operation**. To do so, we assume that the current content of the data subset $DS(i)$ is suitably representative of the classification task. That is to say, it is only the content of DS that is labelled, and the content is incrementally updated from each SW location with the data archiving policy enforcing a finite archive size (Section 3.2). A metric is now necessary for identifying the champion individual relative to the GP individuals identified as *non-dominated*. In limiting the available candidate GP classifiers to the non-dominated set, we reduce the likelihood of selecting degenerate classifiers. Given that class balance is not enforced on $SW(i)$ content, it is desirable to assume a metric that is robust to class imbalance (skew). With this in mind the following definition for average detection rate is assumed:

$$\begin{aligned} DR &= \frac{1}{C} \sum_{c=[1,\ldots,C]} DR_c \\ DR_c &= \frac{tp_c}{tp_c + fn_c} \end{aligned} \tag{1}$$

where $C$ is the number of classes observed in the dataset so far and $tp_c$ and $fn_c$ denote true positive and false negative counts w.r.t. class $c$ respectively.

## 3.4 Symbiotic Bid-Based (SBB) GP

Symbiotic Bid-Based GP, or SBB for short, is a generic coevolutionary GP framework originally developed to facilitate task decomposition under discrete decision making tasks [29]. SBB has been applied to a wide range of problem categories such as reinforcement learning (e.g. [15]) and classification (e.g. [16]).

SBB maintains two populations: symbiont and host. Symbionts ($sym$) take the form of bid-based GP individuals [29]. They specify a 'program' ($p$) and a scalar 'action' ($c$). The action in a classification context takes the form of a class label, assigned when each program is initialized. Individuals from the host population index some subset of the individuals from the symbiont population under a variable length representation.

Host ($h$) evaluation w.r.t. training exemplar ($x$) involves executing the program of each of the symbionts associated with that host, and identifying the bid-based GP with highest output, or:

$$sym^* = \arg\max_{sym \in h} \Big( sym(p, x) \Big) \tag{2}$$

Table 1: Benchmarking dataset properties. $D$ denotes dimensionality, $N$ refers to the total exemplar count, and $k$ is the number of classes

| Stream / Datadataset-set | D | N | $k$ | $\approx$ Class Distribution (%) |
|---|---|---|---|---|
| Gradual Concept Drift (drift) | 10 | 150,000 | 3 | [16, 74, 10] |
| Sudden Concept Shift (shift) | 6 | 6,500,000 | 5 | [37, 25, 24, 9, 4] |
| Electricity Demand (elec) | 8 | 45,312 | 2 | [58, 42] |
| Forest Cover Types (cover) | 54 | 581,012 | 7 | [36, 49, 6, 0.5, 1.5, 3, 4] |

This 'winning' bid-based GP individual ($sym^*$) suggests its corresponding action or class label $sym^*(c)$. The only constraint on host membership is that there must be at least two symbionts with different class labels per host, or:

$$\forall\ sym_{i,j} \in h;\ \ \exists\ sym_i(c) \neq sym_j(c) \qquad (3)$$

where $i$, $j$ are symbiont indexes and $i \neq j$. Hence, multiple symbionts might co-operate to represent a single class. Moreover, previous research with SBB under streaming tasks (without label budgets) indicated that class membership could be incrementally evolved over the course of a stream [2, 1]. This incremental evolution of class membership avoids the requirement for teams to solve all aspects of the class assignment task simultaneously.

Variation and selection operators remain unchanged from the original formulation of SBB [29, 16], and without loss of generality the form of GP assumed for symbiont programs is that of linear GP. The instruction set includes: $\{+, -, \times, \div, \cos(\cdot), \exp(\cdot), \log(\cdot)\}$, although others can be added. Readers are referred to the earlier SBB papers for further details of operators and instruction set of SBB [29, 16].

## 4 Experimental Methodology

This section begins by establishing the approach to benchmark dataset selection (Section 4.1). Section 4.2 discusses parameter setting and characterizes StreamSBB design decisions. Section 4.3.1 outlines the approach adopted to performance evaluation. Finally, the properties of an alternate streaming classifier (an adaptive form of Naive Bayes with label budgeting) is summarized in Section 4.3.2.

### 4.1 Streams / datasets

Four streams / datasets will be employed for the purposes of benchmarking: 1) two artificially created and therefore with known degrees of non-stationary behaviour;[1] "Gradual Concept Drift" and "Sudden Concept Shift" streams, and; 2) two well known real world datasets; "Electricity Demand" [23], and "Forest Cover Types" [3] that have frequently been employed for streaming data benchmarking tasks. The basic properties of the datasets are summarized by Table 1.

[1]Publicly available at `http://web.cs.dal.ca/~mheywood/Code/SBB/Stream/StreamData.html`

**Gradual Concept Drift stream** [17]: Hyperplanes are defined in a 10-dimensional space. Initial values of the hyperplane parameters are selected with uniform probability. This Dataset has 150,000 exemplars and every $1,000$ exemplars, half of the parameters are considered for modification with a 20% chance of change, hence creating the gradual drift of class concepts. Class labels are allocated as a function of hyperplanes exceeding a class threshold.

**Sudden Concept Shift stream** [44]: The Dataset Generator tool[2] is used to construct decision trees that specify a partitioning of the attribute space into a 5-class classification task based on randomly generated thresholds. Data is generated with a uniform p.d.f. and then assigned a class using the decision tree. A total of two concept generator decision trees (C1, C2) are used to create two sources of data. A single stream of data is then constructed block-wise with data integrated from each of the two concept generator decision trees.

The process used to create sudden changes in the concept of classes of the stream has the following form. The stream is created 'block-wise' with 13 blocks and each block consists of 500,000 exemplars. Consider a concept generator tuple of the form: $\langle C1\%, C2\% \rangle$. We can now define the stream interns of the transition of exemplars from 100% $C1$ to 100% $C2$ in 10% increments: $\langle 100, 0 \rangle$, $\langle 100, 0 \rangle$, $\langle 100, 0 \rangle$, $\langle 90, 10 \rangle$, $\langle 80, 20 \rangle$, ... $\langle 0, 100 \rangle$. For example, $\langle 80, 20 \rangle$ denotes a block consisting of exemplars in which 80% are from $C1$ and 20% are from $C2$. A uniform probability is used to determine the exemplar order in each block.

**Electricity Demand** characterizes the rise and fall of electricity demand in New South Wales, Australia, using consumption and price information for the target and neighbouring regions [23]. As such it is a two class dataset (demand will either increase or decrease relative to the previous period), moreover, unlike the other three datasets the distribution of classes is almost balanced.

**Forest Cover Types** defines forest cover type from cartographic variables [3]. The actual forest cover type for a given observation ($30 \times 30$ meter cell) was determined from US Forest Service (USFS) Region 2 Resource Information System (RIS) data. Forest cover type is a 7-class dataset with 54 attributes, 44 of which are binary. The distribution of classes are very imbalanced with the largest class covering almost 50% of dataset and the smallest class covering a mere 0.5%, almost $\frac{1}{100}$ of the majority class. In order to provide a temporal property, previous research sorted the dataset based on the elevation of the $30 \times 30$ meter cells to give it characteristics of streaming data [39]. The same approach is adopted in this paper. What makes this dataset interesting from a streaming data perspective is that there are a comparatively large number of classes, and the seventh class does not appear until roughly half way through the stream. Thus, any classifier working under a label budget would need to discover the new class and react accordingly without disrupting its performance on the other six classes.

## 4.2 Parameterization of GP

Relative to the earlier work [2, 1, 40], the following represents a much more through experimental evaluation. Indeed, the StreamSBB framework of Figure 1 was only proposed in [40] and then benchmarked under a very restrictive scenario (i.e. limited to choices for the label budget). In this work, the objective is to identify what properties of StreamSBB contribute to specific capabilities under streaming data with non-stationary properties. With this in mind, the following three generic parameterizations will be assumed through out:

**Model initialization** is performed using the first $S_{init}$% of the stream during the first $i_{init}$% of generations. This represents the **pre-training budget**, with the remainder of the

[2]Gabor Melli. The 'datgen' Dataset Generator. `http://www.datsetgenerator.com/`

Table 2: Stream / datasets with different generation count and label budgets ($LB$)

| Stream / Dataset | $S_{max}$ | $i_{max}$ | $LB$ | $i_{max}$ | $LB$ |
|---|---|---|---|---|---|
| Gradual Concept Drift (drift) | 150,000 | 500 | 6.7% | 1,000 | 13.3% |
| Sudden Concept Shift (shift) | 6,500,000 | 1,000 | 0.3% | 10,000 | 3.1% |
| Electricity Demand (elec) | 45,312 | 500 | 22.1% | 1,000 | 44.2% |
| Forest Cover Types (cover) | 581,012 | 1,000 | 3.4% | 10,000 | 34.4% |

label budget being consumed across the remainder of the stream. Given that the interface to the stream assumed by StreamSBB is a non-overlapping window, then this assumption just defines the initial window length and assumes that $i_{init}\%$ of the generations are performed against this window location. Thereafter, the sliding window advances at a fixed rate through the stream.

A non-overlapping **sliding window** of length $S_{max}/i_{max}$ exemplars is assumed after model initialization. The remainder of the stream passes through at a constant rate. The window content defines the pool from which the new $|Gap(i)|$ training exemplars are sampled and labels requested (Figure 1). This results in a 'non-overlapping' sliding window. However, GP is evaluated w.r.t. the content of the Data Subset, $|DS|$, (Figure 1), and only $|Gap|$ exemplars are introduced per window location, hence, there is still a 'gentle' turnover in new to old exemplar content between consecutive generations. Parameters are set to $|Gap| = 20$ and $|DS| = 120$ in all experiments.

**Label budget** is the ratio of points whose labels are requested to the total stream length, or:

$$label\ budget(LB) = \frac{i_{max} \times |Gap|}{S_{max}} \tag{4}$$

In other words only $i_{max} \times |Gap|$ exemplars are requested for their label in a stream of length $S_{max}(\equiv N)$. Under the sudden concept shift stream with $S_{max} = 6,500,000$ exemplars and $i_{max} = 1,000$ generations, the non-overlapping window defines 6,500 exemplars between updates of the GP population. In the parameterization assumed here only $|Gap| = 20$ exemplars are added to $DS(i)$ (by the Sampling Policy) at each generation. Hence, the label budget in this example would be:

$$\frac{1,000 \times 20}{6,500,000} \approx 0.3\%$$

Given the rather different stream lengths of the benchmarking datasets (Table 1), different parameterizations for $i_{max}$ will be assumed per dataset as follows:

**Case 1** For Concept Drift and Electricity Demand streams: $i_{max} \in \{500; 1,000\}$

**Case 2** For Concept Shift and Forest Cover Type streams: $i_{max} \in \{1,000; 10,000\}$

This defines two label budgets (LB) per dataset, as summarized by Table 2. Note that $i_{max}$ is taken to include the pre-training budget $i_{init}\%$.

Table 3 summarizes the remaining generic SBB parameter settings assumed in this study e.g., population size, variation and selection operator frequencies. The following new questions are addressed to illustrate the role of design decisions made during StreamSBB and have not previously been considered:

Table 3: Generic SBB parameters. Symbiont population varies dynamically, hence no size parameter is defined. SBB assumes a 'breeder' model of evolution in which $M_{gap}$ hosts are removed per generation [16].

| Parameter | Value |
|---|---|
| DS size ($DS$) | 120 |
| Host population size ($M_{size}$) | 120 |
| Probability of symbiont deletion ($p_d$) | 0.7 |
| Probability of symbiont addition ($p_a$) | 0.7 |
| Probability of action mutation ($\mu_a$) | 0.1 |
| Maximum symbionts per host ($\omega$) | 20 |
| DS gap size ($Gap$) | 20 |
| Host population gap size ($M_{gap}$) | 60 |

Let the initial period of fixed sliding window over 10% of stream (during which the model is constructed) be called 'pre-training' stage and the period of non-overlapping sliding window over $S_{max}/i_{max}$ exemplars be called 'post-training' stage. Two **pre-training bias** experiments are considered in an effort to assess the impact of initial model construction on performance attained across the rest of the stream. In effect we are asking whether spending more resource on the pre-training period has a negative impact on the ability to react to the content in the post-training period (remainder of the stream). Specifically, we are interested in the impact of: 1) longer model construction time with same percentage of exemplar labels, and 2) more label budget during model construction period with the same amount of time.

**DS oversampling** reflects the ability of StreamSBB to decouple the rate at which GP training epochs are performed from the rate at which the data subset content is updated. Specifically, updates to data subset ($DS$) and window location ($SW$) are synchronized and therefore assume the same index, $i$ (Figure 1). Conversely, for each $DS(i)$ we consider the case of performing multiple training epochs, or $j = i \times k$ where $k = 1$ implies one GP generation per $DS(i)$, whereas a parameterization of $k = 5$ implies five GP generations per $DS(i)$. Decoupling GP training epochs in this way potentially provides SBB more time to learn form each data subset. Note also that this does not change the label budget.

**Monolithic vs. modular models:** The original StreamSBB is allowed to represent/model each class label using an evolved mix of symbionts (programs). This implies that multiple programs might coevolve to represent the same class label (task decomposition) [30, 16]. We are interested in investigating the contribution of such open-ended modularity under the context of non-stationary streaming tasks. To do so, we introduce a constrained version of StreamSBB in which a host cannot have more than one symbiont (program) with the same action (class label). All other properties are unchanged; hereafter this will be referred to as the monolithic model.

## 4.3 Evaluation

Two performance metrics will be adopted for characterizing performance: a "prequential accuracy" and "incremental class-wise detection rate". Such metrics are applied relative to the champion individual assumed for labelling stream content. Likewise two comparator classifiers are assumed for the purpose of comparison: a "no-change" model [7] and an Adaptive Naive Bayes classifier with fixed label budgeting [39]. A summary of each follows:

### 4.3.1 Performance Metrics

**Prequential accuracy** [11] represents the most widely used performance metric for streaming data benchmarking. Specifically, the prequential accuracy at exemplar $t$ in the stream is 'weighted' relative to all past $t-1$ exemplars as well as exemplar $t$, or

$$preq_t = \frac{(t-1) \times preq_{t-1} + R_t}{t} \tag{5}$$

where $R_t = 1$ denotes a correct classification of exemplar $t$, and $R_t = 0$ denotes otherwise. The ratio of time indexes acts as a weighting factor, enforcing a decay for older updates [22]. The resulting prequential accuracy takes the form of a curve, although current benchmarking practice also tends to emphasize the reporting of the final prequential accuracy estimate for $t = S_{max}$ as the indication of overall model quality.

A second performance metric, **incremental class-wise detection rate** will also be assumed. The basic motivation is to reduce the sensitivity of the performance metric to class imbalance. This is particularly important under streaming data situations as models are updated incrementally and therefore sensitive to the distribution of current window content (typically a skewed distribution of classes even when the overall class distribution is balanced). The incremental class-wise detection rate can be estimated directly from stream content as follows:

$$\begin{aligned} DR(t) &= \frac{1}{C} \sum_{c=[1,\ldots,C]} DR_c(t) \\ DR_c(t) &= \frac{tp_c(t)}{tp_c(t) + fn_c(t)} \end{aligned} \tag{6}$$

where $t$ is the exemplar index, $tp_c(t)$, $fn_c(t)$ are the respective running totals for true positive and false negative rates up to this point in the stream.

### 4.3.2 Comparator Models

The **No-Change Classifier** requires complete label information, but represents a naive 'devils advocate' solution. The no-change 'classifier' is actually a 1-bit finite state machine in which the state is seeded by the class label, $l(t) = c$, for the present exemplar, $\vec{x}(t)$. The state machine 'predicts' this class for the next exemplar(s) until there is a change in the exemplar class label. A change in the label for exemplar $\vec{x}(t+n)$ to $l(t+n) \neq c$ results in a change in the 'prediction' for exemplar $\vec{x}(t+n+1)$ to that of the new class, say, $c'$. The process then repeats with each change in class label for the current exemplar being assumed as the prediction for the next exemplar. Such a predictor achieves a high accuracy when there are continuous sequences of exemplars in the stream with the same label. Naturally, such a no-change classifier provides a 'feel' for how much implicit class variation exists in a stream.

The second comparator classifier is documented in a recent study of streaming data classification under label budgets and drift detection [39], and has been made available in the Massive Online Analysis (MOA) toolbox.[3] Specifically, the **Naive Bayes** classifier with budgeted active learning and drift detection under the prequential evaluation task is employed. The drift detection mechanism selected was the DDM (Drift Detection Method) algorithm from [21] with default values for threshold (1) and step parameters (0.01). The 'random' active learning strategy was selected as it provided the baseline in [39] and is closest to the stochastic sampling policy adopted in this work. Finally the label budget

[3]MOA prerelease 2014.03; `http://moa.cms.waikato.ac.nz/overview/`

parameter was selected according to the label budget settings of the StreamSBB method. Thus, a new classifier is built when the current classifier's performance begins to degrade, i.e. the current classifier is replaced when drift is explicitly detected by the DDM. Active learning with budgeting is managed under a random exemplar selection policy in which stream data is queried for labels with frequency set by the budget parameter. We also note that both the Naive Bayes and the StreamSBB classifier are configured to label exemplars based on each exemplar instance. Thus, no use is made of features designed to represent temporal properties such as tapped delay lines (see [40] for StreamSBB configured under this scenario).

## 5 Results

Section 4.2 discussed configuration of StreamSBB in terms of the generic parameterization, and higher level design decisions. We start by adopting the basic parameterization decisions for the duration of pre-training, label budget, sliding window size, gap size and illustrate the utility of the stream performance metrics (Section 4.3.1). A common minimal label parameterization is then adopted to enable us to review the impact of making higher level design decisions regarding: model initialization, oversampling, and support for modularity in GP. Having established a preferred set of design decisions, we then compare against the Adaptive Naive Bayesian classifier from the MOA toolbox.

**Model initialization and label budget:** Figures 2, 3, 4, and 5 provide a behavioural summary of StreamSBB in terms of prequential accuracy (Equ. (5)) and incremental detection rate (Equ. (6)) w.r.t. different labelling budgets (Table 2) over the four datasets. In all cases each curve is the result of averaging the performance of GP over 50 runs for each configuration. A common parameterization will be assumed throughout for the generic GP parameters of StreamSBB (Table 3). Given that the sliding window follows a non-overlapping definition, then the size of the window is effectively parameterized by the label budget and corresponding frequency of gap sampling (Table 2). Likewise, the number of labels per sliding window location is fixed ($|Gap| = 20$) for a data subset of $|DS| = 120$. Moreover, at this point we will not consider the impact of more advanced features (oversampling etc), thus one training epoch is performed per window location.

As mentioned during Section 4.2, the first 10% of the stream is made available for initial model construction.[4] The performance curves reflect the operation of the champion classifier (Section 3.3) as the stream data passes. Insight into the degree of mixing of class labels as the stream progresses is provided by the 'no-change' classifier curve (black solid curve). Thus, the artificial **drift** dataset begins with continuous sequences of the same class and then experiences a 20% reduction as the stream progresses (Figure 2). However, the artificial **shift** dataset experiences a high degree of mixing of class label throughout (Figure 3). As previously been pointed out [7], the **electricity** dataset has a low degree of mixing (Figure 4) whereas the **cover type** dataset sees both periods of variation and continuity in the label during the stream (Figure 5).

During the artificial concept **drift** dataset, steady gradual improvements are made throughout the course of the stream, resulting in performance eventually surpassing / reaching that of the no-change classifier under both metrics. Note that the no-change classifier performance is always described in terms of accuracy. The decaying trend of the no-change classifier implies that label mixing increases as the stream progresses.

Under the artificial **shift** dataset, there is insufficient continuity in labels for the no-change classifier to approach the performance of StreamSBB. Note also that under the shift

[4]Implying that 10% of the label budget is consumed in pre-training.

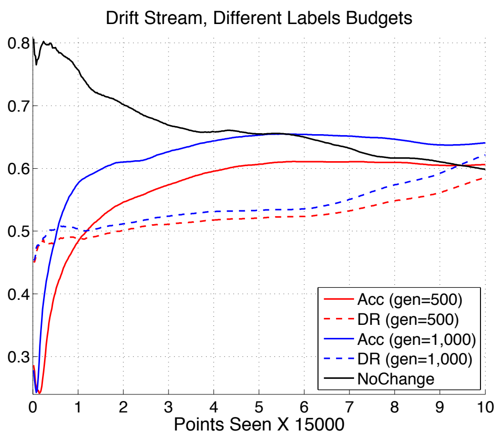

Figure 2: StreamSBB on gradual concept **drift** stream. Curve of GP accuracy (solid) and DR (dash) during stream. First 10% of stream ($1.5 \times 10^4$ exemplars) are used to construct the initial model. Red: Label budget $\approx$ 6.7% or $i_{max} = 500$; Blue: Label budget of $\approx$ 13.3% or $i_{max} = 1,000$; Black: No-Change model.

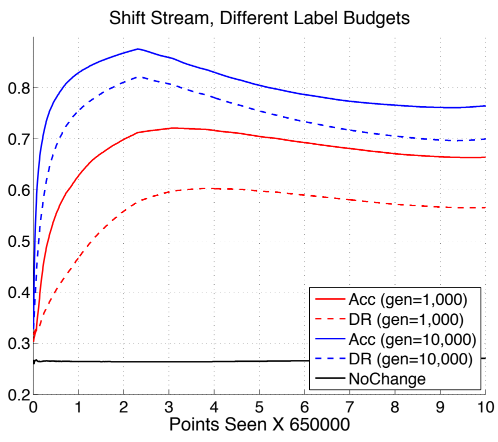

Figure 3: StreamSBB on sudden concept **shift** stream. Curve of GP accuracy (solid) and DR (dash) during stream. First 10% of stream ($6.5 \times 10^5$ exemplars) are used to construct the initial model. Red: Label budget $\approx$ 0.3% or $i_{max} = 1,000$; Blue: Label budget of $\approx$ 3.1% or $i_{max} = 10,000$; Black: No-Change model.

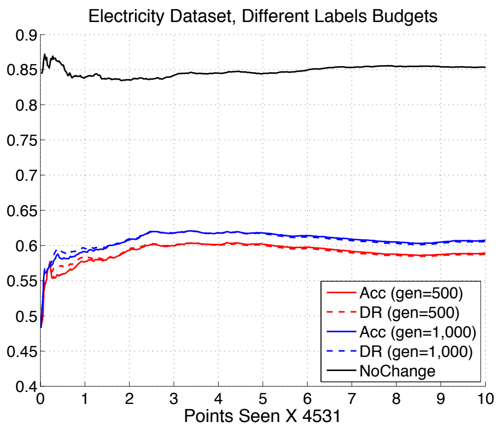

Figure 4: StreamSBB on **electricity** dataset. Curve of GP accuracy (solid) and DR (dash) during stream. First 10% of stream ($4.5 \times 10^3$ exemplars) are used to construct the initial model. Red: Label budget $\approx$ 22.1% or $i_{max} = 500$; Blue: Label budget of $\approx$ 44.2% or $i_{max} = 1,000$; Black: No-Change model.

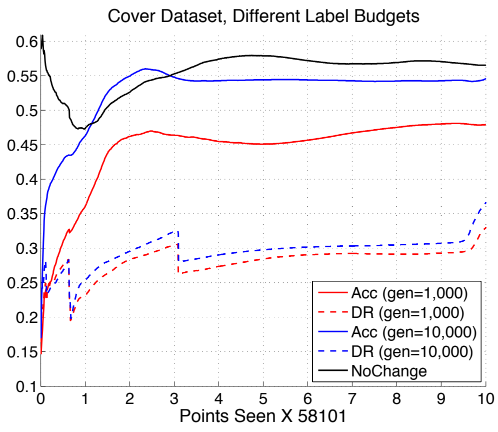

Figure 5: StreamSBB on **cover type** dataset. Curve of GP accuracy (solid) and DR (dash) during stream. First 10% of stream ($5.8 \times 10^4$ exemplars) are used to construct the initial model. Red: Label budget $\approx$ 3.4% or $i_{max} = 1,000$; Blue: Label budget of $\approx$ 34.4% or $i_{max} = 10,000$; Black: No-Change model.

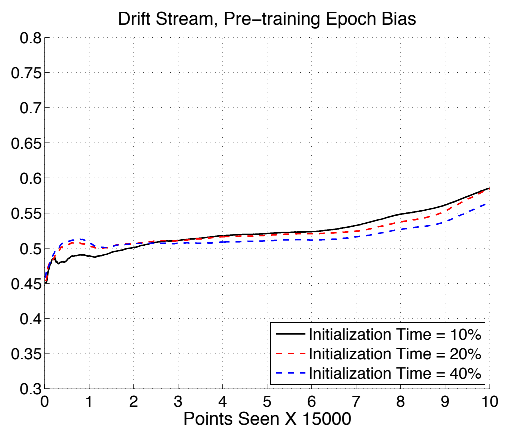

Figure 6: Pre-training epoch bias experiment. *preqDR* on **drift** stream. Black: default 10% (case 1); Red: 20% (case 2); and Blue: 40% (case 3).

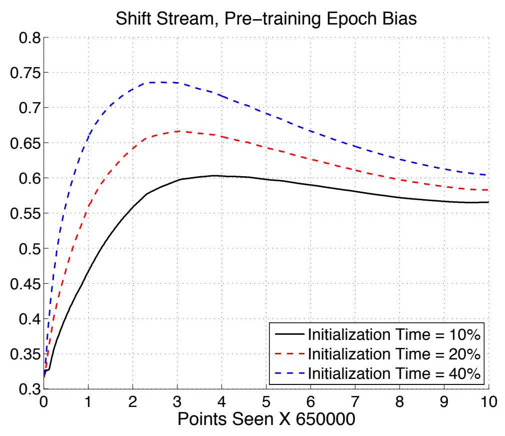

Figure 7: Pre-training epoch bias experiment. *preqDR* on **shift** stream. Black: default 10% (case 1); Red: 20% (case 2); and Blue: 40% (case 3).

dataset, the first 1,500,000 exemplars of the stream are drawn from concept $C1$. This lasts for nearly twice as long as the initial period of pre-training. Thus, as the concept generating the 5 classes shifts from $C1$ to $C2$ during the remaining course of the stream, a decay of $\approx$ 10% in either metric appears irrespective of total label budget.

The accuracy and detection rate curves for GP are almost identical for the **electricity** demand dataset and quickly reach a plateau under this configuration of StreamSBB (Figure 4). Performance of the no-change classifier benefits from continuous sequences of exemplars carrying the same class label. Note that the electricity dataset is a 2 class dataset with 58% to 42% class distribution (almost balanced). Finally, the **forest cover type** dataset illustrates several dynamic properties. Pre-training only results in 6 of 7 classes appearing. Thus, when instances of the 7th class do appear (approximately where index 3 appears in the $x$-axis), then there is a corresponding drop in detection rate (detection rate having been estimated over 6 classes up to this point). There are also several transient properties earlier in the stream, which appear to be indicative of sudden context switches in the underlying process as they impact both metrics and forms of classifier.

It is also evident that the accuracy metric is strongly biased by the class distribution, resulting in an 'over optimistic' performance curve in all but the balanced data set (electricity); a property widely observed under non-streaming classification benchmarks. With this in mind, we will adopt the detection rate metric in the remainder of the study.

**Pre-training epoch bias experiment:** Figures 6 and 7 illustrate the impact of varying the distribution of StreamSBB training epochs between pre-training and the remainder of the stream. Note that previously the pre-training period (performed against 10% of the data) consumed an equal amount of the total training epochs (10%). In the case of these experiments, pre-training is still performed against 10% of the data (thus still only utilizing a 10% label budget), but consumes 20% or 40% of the training epochs. Naturally, the total number of training epochs per stream is unchanged, leading to the following three configurations:

**Case 1** The default case, i.e. 10% of evolution time is dedicated to model construction and 90% after.

**Case 2** 20% of evolution time is dedicated to model construction and 80% after.

**Case 3** 40% of evolution time is dedicated to model construction and 60% after.

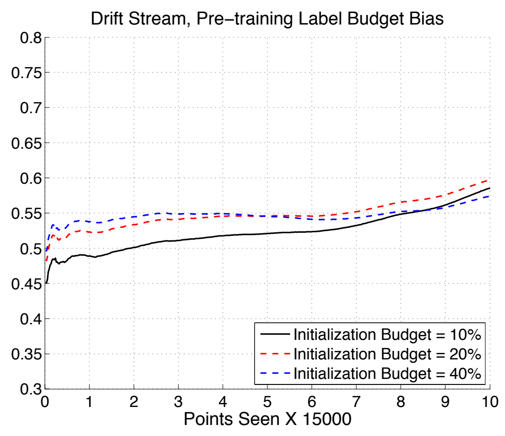

Figure 8: Pre-training label budget bias experiment. *preqDR* on **drift** stream. Black: default 10% (case 1); Red: 20% (case 2); and Blue: 40% (case 3).

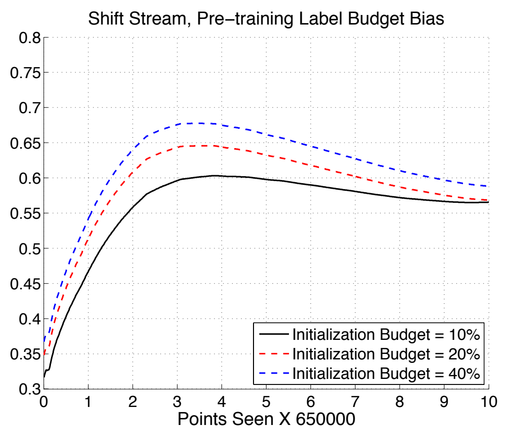

Figure 9: Pre-training label budget bias experiment. *preqDR* on **shift** stream. Black: default 10% (case 1); Red: 20% (case 2); and Blue: 40% (case 3).

It appears that there is no lasting benefit to be gained from biasing more training epochs to the pre-training period. Under the **shift** dataset (Figure 7), a significant regression back to the vicinity of 'case 1' detection rate appears by the end of the stream. In short, the benefit gained by introducing biases towards more pre-training time are lost over the remainder of the stream.[5]

**Pre-training label budget bias experiment:** Here we ask wether using more of the label budget during pre-training will provide the basis for better models during the remainder of the stream. Specifically, rather than providing more time to the pre-training period we provide a larger proportion of label budget to the pre-training period. Note that the overall label budget is intact, however more labels are requested during model construction and less thereafter. This leads to the following three configurations:

**Case 1** The default case of uniform sampling through the stream, i.e. 10% of label and training budget is requested during construction and 90% after.

**Case 2** 20% of label and training budget is requested during model construction and 80% after.

**Case 3** 40% of label and training budget is requested during model construction and 60% after.

Figures 8 and 9 summarize the impact of pre-training label budget bias on detection rate for the concept **drift** and **shift** datasets respectively. In both cases the results are very similar to those observed under the bias to training epochs.[6] Any improvement relative to the drift data set (compare Figure 8 to 6) being lost under the shift data set (compare Figure 9 to 7). In short, no real benefit is observed in biasing more training generations or label budget towards the initial pre-training period.

**DS Oversampling experiment:** Sections 3 and 4.2 made the case for relaxing the relation between updates to the data subset ($DS(i)$) and performing a training epoch ($Gen(j)$, Figure 1). Thus, for each update to the data subset rather than conduct a single generation,

[5]Electricity demand and forest cover type datasets observed similar effects and therefore results are not explicitly reported.

[6]Similar effects being observed for the electricity and forest cover type datasets.

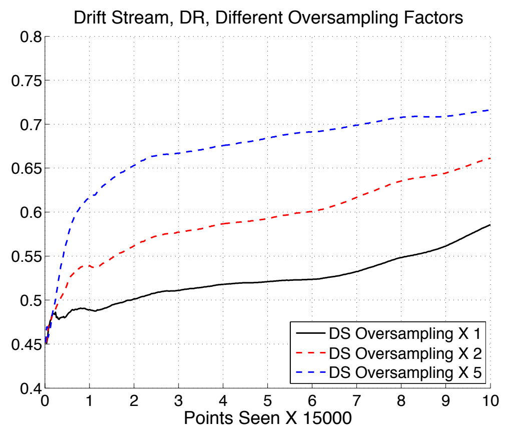

Figure 10: Oversampling experiment. *preqDR* on **drift** stream. Black: default sampling; Red: $\times 2$ oversampling; and Blue: $\times 5$ oversampling.

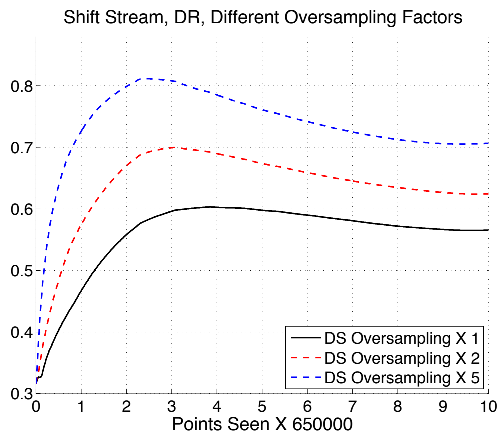

Figure 11: Oversampling experiment. *preqDR* on **shift** stream. Black: default sampling; Red: $\times 2$ oversampling; and Blue: $\times 5$ oversampling.

multiple generations might be performed. This does not change the label budget and does not represent a bias towards pre-training as it is performed throughout the whole stream. Three parameterizations are considered:

**Case 1** The default case of uniform sampling through the stream.

**Case 2** DS oversampling by a factor of 2.

**Case 3** DS oversampling by a factor of 5.

Figures 10 and 11 illustrate the impact of oversampling in terms of incremental detection rate curves for concept **drift** and **shift** streams. Higher detection rates are now maintained throughout the stream. Indeed, the higher rate of oversampling appears to be preferable throughout, although further increases to the oversampling (a factor of 10) only had marginal effects compared to the case of oversampling to a factor of 5 (overlearning). Results for electricity and cover type were also positive and will be reported later when we compare with the Adaptive Naive Bayesian framework for streaming classification.

**Monolithic vs. modular:** In all the previous experiments StreamSBB was used in its original modular configuration, i.e. the number of symbionts per class labels were allowed to freely evolve. Previous benchmarking performed under a classical non-streaming setting of classification through supervised learning indicated that such open-ended evolution of modularity was particularly beneficial [30]. Other researchers have also noted that the open-ended evolution of modularity can be potentially beneficial in 'dynamic' environments (Section 2). In this experiment we compare StreamSBB with a modified version in which there can only be a single program per team per class. Although, still modular at some level (there are as many programs as classes) we will refer to this as a monolithic classifier, and StreamSBB as a modular classifier.[7] The objective of this experiment is to quantify to what extent support for such open-ended evolution of modularity is beneficial under non-stationary streaming classification tasks.

Figures 12 and 13, summarize detection rate on the concept **drift** and **shift** streams respectively. There is a statistically significant difference in favour of assuming open-ended

[7]Other than the monolithic formulation of SBB being subject to the constraint that only one program may represent each class, the two implementations are the same.

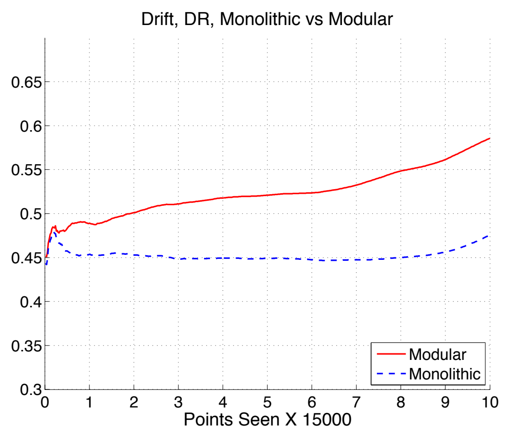

Figure 12: Detection rate of concept **drift** stream under modular (solid) vs. monolithic (dash) configurations.

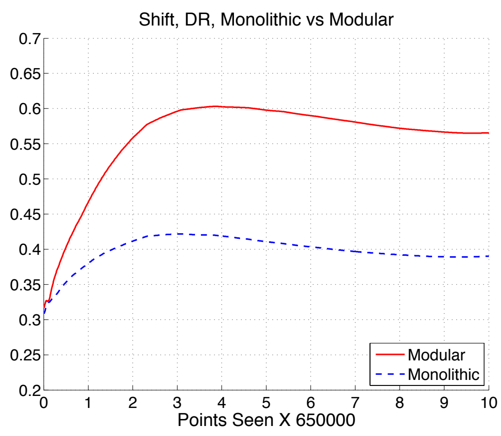

Figure 13: Detection rate of concept **shift** stream under modular (solid) vs. monolithic (dash) configurations.

evolution of modularity (Student T-test $p$-value of $3.14 \times 10^{-237}$ and 0 for concept drift and concept shift streams respectively under 0.01 significance level). Thus, modularity is synonymous with task decomposition, which under domains that undergo change potentially implies that only a subset of programs within a modular solution need to be revised when a change occurs. Conversely, under monolithic solutions, it is more difficult to explicitly delimit what variation operators modify, hence identifying the relevant parts of a program to modify becomes more difficult.

Under non-stationary streams, given that change takes place throughout the stream we can also review the development of age of the champion individual through the course of the stream. Note that the champion solution is the individual used to *provide* labels as the stream progresses, with identification of the champion performed relative to the current content of the data subset (Section 3.3). Naturally, each time the data subset is updated, the champion might change. Age is the number of training epochs for which an individual manages to exist.

Figures 14 and 15 illustrate the average age of the champion for concept **drift** and **shift** streams respectively. Both plots suggest that the average age of champion hosts of the modular configuration (red curve) remains lower throughout the stream. In effect, there is a higher rate of turn over of champion individuals when modularity is supported (implied by the lower age of champions). This reflects a stronger ability to react to change. Conversely, the much higher age of champions under the monolithic framework appears to indicate that the same champion has to be used for longer before better replacements are found. This has obvious decremental consequences for classifier performance.

**StreamSBB vs. Adaptive Naive Bayes:** Results for the second baseline classifier, the Adaptive Naive Bayesian (ANB) framework for streaming data classification under label budgets as implemented in the MOA toolkit (Section 4.3.2) are summarized in Figures 16 through 19. Detection rate of StreamSBB and ANB models with similar label budgets are compared against each other for the 4 datasets.

StreamSBB results are presented in terms of a set of curves illustrating the impact of assuming different DS oversampling rates, i.e. the number of training epochs performed per DS update. As noted in the earlier experiments this appears to be the most important design decision and has no impact on the label budget. In all cases the solid black curve is the incremental detection rate for ANB. Under the **drift** stream, ANB appears to take longer

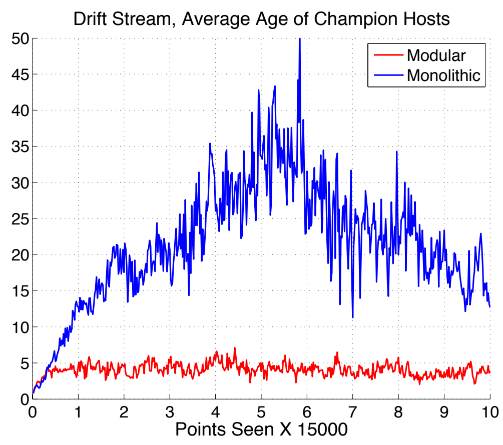

Figure 14: Average age of the champion individual during evolutionary loop, concept **drift** stream.

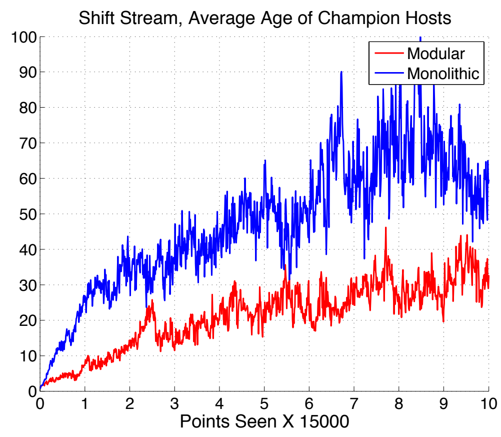

Figure 15: Average age of the champion individual during evolutionary loop, concept **shift** stream.

to develop an initial classifier. Thereafter, both models alternate until the end of stream where they settle on the same detection rate. Conversely, ANB appears to over learn the initial configuration of the **shift** stream (corresponding to concept C1), and then lose 25% of its initial detection rate (going from 80% to 55%) over the remaining 75% of the stream. Conversely, StreamSBB experiences significantly less loss over the course of concept $C2$ being introduced during the last three quarters of the stream (Figure 17).

The **electricity** dataset results in StreamSBB returning a constant detection rate of 58 to 60% throughout the stream, but never approaching the performance returned by ANB (Figure 18). The behaviour under the **covertype** dataset is more interesting. During the first three intervals of the stream, both models undergo sharp changes in the detection rate (Figure 19). A sudden drop then appears as the seventh class is encountered for the first time, and therefore all models miss-classify this class (see $x$-axis value $\approx$ 3). The ensuing gradual recovery of the detection rate undergoes a final jump in the last 20 000 exemplars of the sequence (last interval of the $x$-axis).

In summary ANB appears to have problems when there are sudden changes to the content of the stream (shift stream), whereas both algorithms are effective under the drift stream. In both real-world datasets, ANB was more effective, however, StreamSBB might well benefit from the use of tapped delay lines when contracting models for such tasks (both models make classification decisions on the basis of a single exemplar). Further research indicates that StreamSBB does indeed benefit considerably from the use of a delay lines on real-world data sets [41].

## 6 Conclusion

A framework for applying GP to streaming data classification tasks under label budgets is presented. To do so, GP is evolved against a data subset. The subset makes use of Pareto archiving policy with diversity / age heuristics to prioritize exemplars for retention beyond the lifetime of the current window to the stream. A simple uniform sampling scheme is assumed for selecting exemplars for labelling. Attempts to introduce more complex sampling policies (such as biasing label requests towards exemplars that have lower confidence

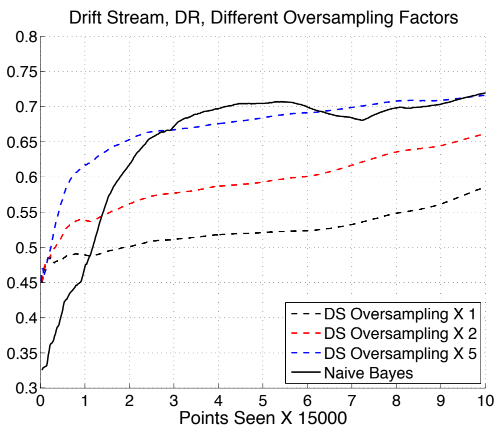

Figure 16: Detection rate of StreamSBB vs. ANB model on concept **drift** stream (label budget = 6.6%).

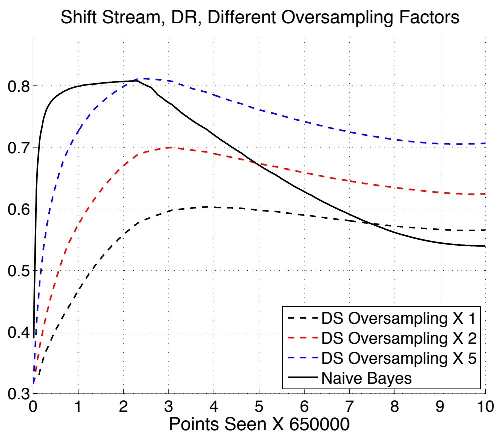

Figure 17: Detection rate of StreamSBB vs. ANB model on concept **shift** stream (label budget = 0.3%).

in classification) generally resulted in worse results than the uniform sampling policy.[8] The data subset was also used as the basis for supporting anytime classifier operation. It is only the data subset that contains labelled exemplars, however, we also make use of Pareto archiving to limit the set of GP individuals to those that are non-dominated (cf. effect of class imbalance on the data subset).

Two factors were identified that had particular significance with respect to StreamSBB performance:

- perform multiple generations per data subset – where this appears to improve the rate of adaptation in GP to updates to the data subset content.
- support for coevolution of programs – where assuming a single (monolithic) program per class resulted in much lower rates of classification than when the number of programs per class was an evolved property.

Benchmarking also introduced a incremental formulation for detection rate where this is more informative of the true classifier behaviour under class imbalance. Future work will assess the utility of tapped delay lines to enable GP to represent temporal properties between sequences of exemplars when labelling exemplar *t*. At present, each exemplar is labelled independently. Moreover, we will continue to extend the set of real-world datasets on which benchmarking is performed.

## Acknowledgement

The authors gratefully acknowledge funding provided by the NSERC CRD grant program (Canada).

[8]Results not shown for brevity.

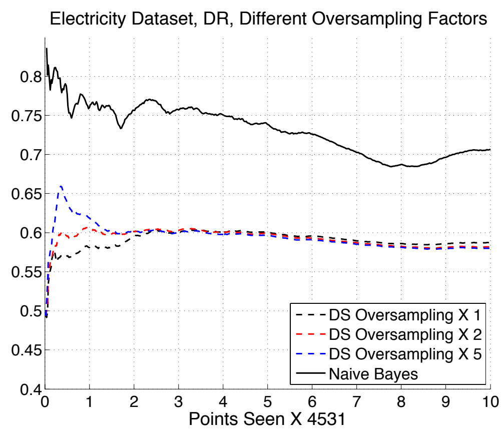

Figure 18: Detection rate of StreamSBB vs. ANB model on **electricity** dataset (label budget = 22.1%).

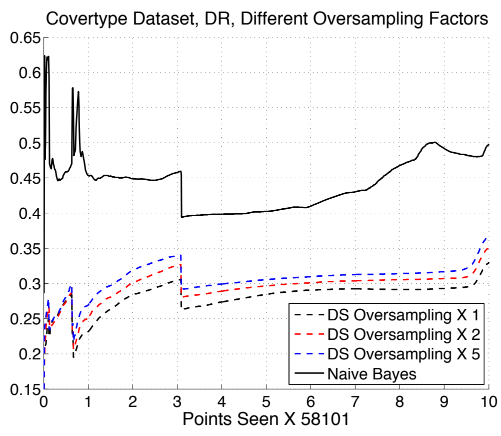

Figure 19: Detection rate of StreamSBB vs. ANB model on **covertype** dataset (label budget = 3.4%).

## References

[1] A. Atwater and M. I. Heywood. Benchmarking Pareto archiving heuristics in the presence of concept drift: Diversity versus age. In *ACM Genetic and Evolutionary Computation Conference*, pages 885–892, 2013.

[2] A. Atwater, M. I. Heywood, and A. N. Zincir-Heywood. GP under streaming data constraints: A case for Pareto archiving? In *ACM Genetic and Evolutionary Computation Conference*, pages 703–710, 2012.

[3] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[4] M. Behdad and T. French. Online learning classifiers in dynamic environments with incomplete feedback. In *IEEE Congress on Evolutionary Computation*, pages 1786–1793, 2013.

[5] A. Bifet. *Adaptive Stream Mining: Pattern Learning and Mining from Evolving Data Streams*, volume 207 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2010.

[6] A. Bifet and R. Gavalda. Learning from time-changing data with adaptive windowing. In *SIAM International Conference on Data Mining*, pages 443–448, 2007.

[7] A. Bifet, I. Žliobaitė, B. Pfahringer, and G. Holmes. Pitfalls in benchmarking data stream classification and how to avoid them. In *Machine Learning and Knowledge Discovery in Databases*, volume 8188 of *LNCS*, pages 465–479, 2013.

[8] T. Blackwell and J. Branke. Multiswarms, exclusion, and anti-convergence in dynamic environments. *IEEE Transactions on Evolutionary Computation*, 10(4):459–472, 2006.

[9] G. Brown and L. I. Kuncheva. “Good” and “bad” diversity in majority vote ensembles. In *Multiple Classifier Systems*, volume 5997 of *LNCS*, pages 124–133, 2010.

[10] T. Dasu, S. Krishnan, S. Venkatasubramanian, and K. Yi. An information-theoretic approach to detecting changes in multi-dimensional data streams. In *Proceedings of the Symposium on the Interface of Statistics*, 2006.

[11] A. P. Dawid. Statistical theory: The prequential approach. *Journal of the Royal Statistical Society-A*, 147:278–292, 1984.

[12] E. D. de Jong. A monotonic archive for pareto-coevolution. *Evolutionary Computation*, 15(1):61–94, 2007.

[13] I. Dempsey, M. O'Neill, and A. Brabazon. *Foundations in Grammatical Evolution for Dynamic Environments*, volume 194 of *Studies in Computational Intelligence*. Springer, 2009.

[14] G. Ditzler and R. Polikar. Hellinger distance based drift detection for non-stationary environments. In *IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments*, pages 41–48, 2011.

[15] J. A. Doucette, P. Lichodzijewski, and M. I. Heywood. Hierarchical task decomposition through symbiosis in reinforcement learning. In *ACM Genetic and Evolutionary Computation Conference*, pages 97–104, 2012.

[16] J. A. Doucette, A. R. McIntyre, P. Lichodzijewski, and M. I. Heywood. Symbiotic coevolutionary genetic programming: a benchmarking study under large attribute spaces. *Genetic Programming and Evolvable Machines*, 13(1), 2012.

[17] W. Fan, Y. Huang, H. Wang, and P. S. Yu. Active mining of data streams. In *Proceedings of SIAM International Conference on Data Mining*, pages 457–461, 2004.

[18] G. Folino and G. Papuzzo. Handling different categories of concept drift in data streams using distributed GP. In *European Conference on Genetic Programming*, volume 6021 of *LNCS*, pages 74–85, 2010.

[19] J. Gama. *Knowledge discovery from data streams*. CRC Press, 2010.

[20] J. Gama. A survey on learning from data streams: Current and future trends. *Progress in Artificial Intelligence*, 1(1):45–55, 2012.

[21] J. Gama, P. Medas, G. Castillo, and P. P. Rodrigues. Learning with drift detection. In *Advances in Artificial Intelligence*, volume 3171 of *LNCS*, pages 66–112, 2004.

[22] J. Gama, R. Sebastião, and P. Rodrigues. On evaluating stream learning algorithms. *Machine Learning*, 90(3):317–346, 2013.

[23] M. Harries. Splice-2 comparative evaluation: Electricity pricing. Technical report, University of New South Wales, 1999.

[24] M. I. Heywood. Evolutionary model building under streaming data for classification tasks: opportunities and challenges. *Genetic Programming and Evolvable Machines*, 2015. DOI `10.1007/s10710-014-9236-y`.

[25] S. Huang and Y. Dong. An active learning system for mining time changing data streams. *Intelligent Data Analysis*, 11(4):401–419, 2007.

[26] N. Kashtan, E. Noor, and U. Alon. Varying environments can speed up evolution. *Proceedings of the National Academy of Sciences*, 104(34):13713–13716, 2007.

[27] D. Kifer, S. Ben-David, and J. Gehrke. Detecting change in data streams. In *Proceedings of the International Conference on Very Large Data Bases*, pages 180–191. Morgan Kaufmann, 2004.

[28] C. Lanquillon. Information filtering in changing domains. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 41–48, 1999.

[29] P. Lichodzijewski and M. I. Heywood. Managing team-based problem solving with Symbiotic Bid-based Genetic Programming. In *ACM Genetic and Evolutionary Computation Conference*, pages 363–370, 2008.

[30] P. Lichodzijewski and M. I. Heywood. Symbiosis, complexification and simplicity under GP. In *ACM Genetic and Evolutionary Computation Conference*, pages 853–860, 2010.

[31] P. Lindstrom, B. MacNamee, and S. J. Delany. Handling concept drift in a text data stream constrained by high labelling cost. In *Proceedings of the International Florida Artificial Intelligence Research Society Conference*. AAAI, 2010.

[32] P. Lindstrom, B. MacNamee, and S. J. Delany. Drift detection using uncertainty distribution divergence. *Evolutionary Intelligence*, 4(1):13–25, 2013.

[33] L. L. Minku, A. P. White, and X. Yao. The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE Transactions on Knowledge and Data Engineering*, 22(5):730–742, 2010.

[34] M. Parter, N. Kashtan, and U. Alon. Facilitated variation: How evolution learns from past environments to generalize to new environments. *PLoS Computational Biology*, 4(11):e1000206, 2008.

[35] J. Quinonero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, editors. *Dataset shift in machine learning*. MIT Press, 2009.

[36] R. Sebastio and J. Gama. Change detection in learning histograms from data streams. In *Proceedings of the Portuguese Conference on Artificial Intelligence*, volume 4874 of *LNCS*, pages 112–123. Springer, 2007.

[37] R. Stapenhurst and G. Brown. Theoretical and empirical analysis of diversity in non-stationary learning. In *IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments*, pages 25–32, 2011.

[38] I. Žliobaitė, A. Bifet, B. Pfahringer, and G. Holmes. Active learning with evolving streaming data. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 597–612. Springer, 2011.

[39] I. Žliobaitė, A. Bifet, B. Pfahringer, and G. Holmes. Active learning with drifting streaming data. *IEEE Transactions on Neural Networks and Learning Systems*, 25(1):27–54, 2014.

[40] A. Vahdat, A. Atwater, A. R. McIntyre, and M. I. Heywood. On the application of GP to streaming data classification tasks with label budgets. In *ACM Genetic and Evolutionary Computation Conference: ECBDL Workshop*, pages 1287–1294, 2014.

[41] A. Vahdat, J. Morgan, A. R. McIntyre, M. I. Heywood, and A. N. Zincir-Heywood. Tapped delay lines for GP streaming data classification with label budgets. In *European Conference on Genetic Programming*, volume 9025 of *LNCS*. Springer, 2015.

[42] P. Vorburger and A. Bernstein. Entropy-based concept shift detection. In *Proceedings of the Sixth International Conference on Data Mining*, pages 1113–1118, 2006.

[43] G. P. Wagner and L. Altenberg. Complex adaptations and the evolution of evolvability. *Complexity*, 50(3):433–452, 1996.

[44] X. Zhu, P. Zhang, X. Lin, and Y. Shi. Active learning from stream data using optimal weight classifier ensemble. *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, 40(6):1607–1621, 2010.